15.07.25 EXPERIMENT-1

1) Write a C++ program to declare a class student having data members as roll_no and name. Accept and display data for single student.

→
```
#include <iostream>
using namespace std;
class student
{ int roll_no;
  string name;
public:
void accept()
{ cout << "Enter the student name and roll no:";
  cin >> name >> roll_no;}
void disp()
{ cout << "student Name:" <<name;
  cout <<'"\n Student Roll No:" <<roll_no;}};
int main()
{ student s1;
  s1.accept();
  s1.disp();
  return 0;}
```

get line

Output:-
Enter the student name and roll no: Kinjal 58
Student Name : Kinjal
Student Roll No : 58

2) Write a C++ code to create a class book

having data members as book_name, book_price and book_id book_pages. Accept the data for 2 books and display the name of the book having greater price.

→
```
#include <iostream>
using namespace std;
class class_book
{string book_name;<get line
 int book_id;
 public:
int book_price;
void accept()
{cout << "Enter the book name, price and ID:";
  cin>>book_name>>book_price>>book_id;}
void disp()
{cout << "\n Book Name:" <<book_name;
 cout<< "\n Book Price:" <<book_price;
 cout<< "\n Book ID:" <<book_id;}};
int main()
{class_book B1, B2;
 B1.accept();
 B2.accept();
 if (B1.book_price > B2.book_price)
 {B1.disp();}
 else
 {B2.disp();}
 return 0;}
```

Output:-
Enter the book name, price and ID: ABC 450 6078
Enter the book name, price and ID: XYZ 599 4067

Book Name: XYZ
Book Price: 599
Book ID: 4067

3) Write a c++ program to declare a class time having data members as H, M and S. Accept data for one object and display total time in seconds.

→
```
#include <iostream>
using namespace std;
class Time
{int H, M, S;
 public:
void accept()
{cout<<"Enter the time in hours minutes seconds:";
 cin>>H>>M>>S;}
void calculate()
{H=H*3600;
 M=M*60;
 S=S+H+M;}
void disp()
```

```
{cout << "Total time in seconds = " << s;}};
int main()
{Time T1;
 T1.accept();
 T1.calculate();
 T1.disp();
 return 0;}
```

Output:-

Enter the time in hours minutes seconds: 2 46 37
Total time in seconds = 9997

15/7

22.07.25

EXPERIMENT-2

1) WAP to declare a class 'city' having data members as name and population. Accept this data for 5 cities and display name of city having highest population.

→
```
#include <iostream>
using namespace std;
class city
{public:
 string name;
 int population;
 void accept()
{cout<<"Enter City Name:";
 cin>>name;
 cout<<"Enter City Population:";
 cin>>population;}
 void disp()
{cout<<"city having highest population:"<<name;}};
int main()
{ city C[5];
 int i, max;
for(i=0; i<5; i++)
{ C[i].accept();}
max=C[0].population;
for(i=0; i<5; i++)
{ if (C[i].population>max)
{ max=i;}}
C[max].disp();
```

return 0;}

Output:-
Enter City Name: Mumbai
Enter City Population: 789
Enter City Name: Pune
Enter City Population: 567
Enter City Name: Jaipur
Enter City Population: 389
Enter City Name: Khasagpur
Enter City Population: 456
Enter City Name: Kolkata
Enter City Population: 987
City having highest population: Kolkata

2) WAP to declare a class 'Account' having da members as Account no. and balance. Acce this data for 10 accounts and give inte of 10% where balance is equal or grea than 5000 and display them.

→ #include <iostream>
using namespace std;
class account
{ public:
int acc_no;
float balance;
void accept()
{ cout << "Enter Account Number: ";

```
cin>> acc_no;
cout<<"Enter Account Balance:";
cin>>balance;}
void disp()
{ cout<<"\nAccount Number:"<<acc_no;
  cout<<"\nAccount Balance:"<<balance;}};
int main()
{ account A[10];
 int i;
for (i=0; i<10; i++)
{A[i].accept();}
for (i=0; i<10; i++)
{if (A[i].balance>=5000)
{A[i].balance =A[i].balance+(0.1* A[i]
 A[i].balance);
 A[i].disp();}}
return 0;}
```

Output:-

```
Enter Account Number: 2564
Enter Account Balance: 2095
Enter Account Balance: 4879
Enter Account Number: 7890
Enter Account Balance: 6789
Enter Account Number: 4567
Enter Account Balance: 3578
Enter Account Number: 2478
Enter Account Balance: 3678
Enter Account Number: 2634
```

Enter Account Balance: 9705
Enter Account Number: 5738
Enter Account Balance: 57228
Enter Account Number: 5835
Enter Account Balance: 3789
Enter Account Number: 4627
Enter Account Balance: 8906
Enter Account Number: 7568
Enter Account Balance: 4689

Account Number: 7890
Account Balance: 7467.9
Account Number: 2634
Account Balance: 10675.5
Account Number: 5738
Account Balance: 62950.8
Account Number: 4627
Account Balance: 9796.6

3) WAP to declare a class 'staff' having data members as name and post. Accept this data for 5 staff and display names of staff who are "HOD".

→
```
#include <iostream>
using namespace std;
class staff
{ string name;
  public:
```

```
string post;
void accept()
{cout<<"Enter the staff name:";
 getline(cin, name);
 cout<<"Enter the staff post:";
 getline(cin, post);}
void disp()
{ cout<<"\nStaff Name:"<<name;
  cout<<"\nStaff Post:"<<post;}};
int main()
{ staff S[5];
 int i;
 for(i=0; i<5; i++)
{S[i].accept();}
for(i=0; i<5; i++)
{ if (S[i].post=="HOD")
 {S[i].disp();}}
 return 0;}
```

Output :-

```
Enter the staff name: Alekha Mishra
Enter the staff post: Lecturer
Enter the staff name: Anishkaa Awasthi
Enter the staff post: HOD
Enter the staff name: Diya Poojary
Enter the staff post: Dean
Enter the staff name: Akanksha Joshi
Enter the staff post : HOD
Enter the staff name: Drushti Kolekar
```

Date
Page 14

Enter the staff post: Principal

Staff Name: Anishkaa Awasthi
Staff Post: HOD
Staff Name: Akanksha Joshi
Staff Post: HOD

5/8

25.07.25

EXPERIMENT-3

1) Write a program to declare a class 'book' containing data members as book_title, author_name and price. Accept and display the information for one object using a pointer to that object.

→
```
#include <stdio.h> <iostream>
using namespace std;
class book
{string booktitle;
 string authorname;
 int price;
 public:
 void accept()
{cout << "Enter Book Title:";
 getline (cin, booktitle);
 cout << "Enter Author name:";
 getline(cin, authorname);
 cout << "Enter Book Price:";
 cin >> price;}
 void disp()
{cout << "Book Title:" << booktitle;
 cout << "\nAuthor Name:" << authorname;
 cout << "\nBook Price:" << price;};
int main()
{book B1;
 book *p;
 p = &B1;
```

```
p→accept();
p→disp();
return 0;}
```

Output:-

Enter Book Title: Atomic Habits
Enter Author name: James Clear
Enter Book price: 599
Book Title: Atomic Habits
Author Name: James Clear
Book Price: 599

2) WAP to declare a class 'student' having data members roll_no and percentage. Using 'this' pointer invoke member functions to accept and display this data for one object of the class.

→
```
#include <iostream>
using namespace std;
class student
{ int roll_no;
 float percentage;
 public:
 void accept()
{cout << "Enter the Student Roll No:";
 cin >> roll_no;
 cout << "Enter Student Percentage:";
 cin >> percentage;}
 void disp()
```

```
{ this -> accept();
  cout << "\nRoll No of the student:" << roll_no;
  cout << "\nPercentage of the student:" <<
  percentage;}};
int main()
{ student S1;
  S1.disp();
  return 0;}
```

Output:-

```
Enter the student Roll No: 58
Enter student Percentage: 92.6

Roll No of the student: 58
Percentage of the student: 92.6
```

3) Write a program to demonstrate the use of nested class.

→

```
#include <iostream>
using namespace std;
class student{
    string name;
    int roll_no;
    public:
    void accept(){
     cout << "Enter the student Name:";
      getline(cin, name);
      cout << "Enter the student Roll No:";
```

```
    cin>>roll_no;
}
class marks{
    public:
    int phy;
    int maths;
    int total_p;
    int total_m;
    void accept(){
    cout<<"Enter the total marks for
        physics:";
        cin>>total_p;
        cout<<"Enter the marks obtained in
        physics:";
        cin>>phy;
        cout<<"Enter the total marks for
        maths:";
        cin>>total_m;
        cout<<"Enter the marks obtained in
        maths:";
        cin>>maths;
    }
    void disp(){
        float total=0;
        float percentage;
        total=total_p+total_m;
        percentage=((phy+maths)/total)*100;
        cout<<"\nPercentage obtained by
        the student="<<percentage<<"%";
```

```
    };
};
int main() {
    student s;
    s.accept();
    student::marks m;
    m.accept();
    m.disp();
    return 0;
}
```

→ Output:-

```
Enter the student Name: Kinjal Kolekar
Enter the student Roll No: 58
Enter the total marks for physics: 100
Enter the marks obtained in physics: 96
Enter the total marks for maths: 100
Enter the marks obtained in maths: 98

Percentage obtained by the student = 97%.
```

58

08.08.2025

EXPERIMENT-4

1) WAP to swap two numbers from same class using object as function argument. Write swap function as member function.

→
```
#include <iostream>
using namespace std;
class numbers {
    private:
    int a,b;
    public:
    int temp;
    void accept() {
        cout<<"Enter the first number:";
        cin>>a;
        cout<<"Enter the second number:";
        cin>>b;
    }
    void disp() {
        cout<<"\nAfter swapping:"<<endl;
        cout<<"First number ="<<a<<endl;
        cout<<"Second number ="<<b;
    }
    void swap(numbers &n){
        n.temp = n.a;
        n.a = n.b;
        n.b = n.temp;
    }
};
```

```
int main() {
    numbers n;
    n.accept();
    n.swap(n);
    n.disp();
    return 0;
}
```

Output:-

Enter the first number: 9
Enter the second number: 2

After swapping:
First number = 2
Second number = 9

2) WAP to swap two numbers from same class using concept of friend function.

→
```
#include <iostream>
using namespace std;
class number{
    private:
    int a,b,temp;
    public:
    void accept(){
        cout<<"Enter the first number:";
        cin>>a;
        cout<<"Enter the second number:";
```

```
    cin>>b;
  }
  void disp(){
    cout<<"\nAfter swapping:"<<endl;
    cout<<"First number="<<a<<endl;
    cout<<"second number="<<b;
  }
  friend void swap(number &);
};
void swap(number &n){
    n.temp=n.a;
    n.a=n.b;
    n.b=n.temp;
}
int main(){
    number n;
    n.accept();
    swap(n);
    n.disp();
    return 0;
}
```

Output:-

```
Enter the first number:7
Enter the second number:4

After swapping:
First number=4
second number=7
```

3) WAP to swap two numbers from different class using friend function.

→
```
#include <iostream>
using namespace std;
class num2;
class num1 {
    private:
    int a;
    public:
    void accept() {
      cout << "Enter the first number: ";
      cin >> a;
    }
    friend void swap(num1 &, num2 &);
};
class num2 {
    private:
    int b;
    public:
    void accept() {
      cout << "Enter the second number: ";
      cin >> b;
    }
    friend void swap(num1 &, num2 &);
};
void swap(num1 &x, num2 &y) {
    int temp;
    temp = x.a;
```

```
    x.a = y.b;
    y.b = temp;
    cout << "\nAfter swapping:" << endl;
    cout << "First number=" << x.a << endl;
    cout << "second number=" << y.b << endl;
}
int main() {
    num1 n1;
    num2 n2;
    n1.accept();
    n2.accept();
    swap(n1, n2);
    return 0;
}
```

Output:-

Enter the first number = 5
Enter the second number = 6

After swapping:
First number = 6
Second number = 5

4) WAP to create two classes Result1 and Result2 which stores the marks of the students. Read the value of marks for both the class objects and compute the average of two results.

→

```
#include <iostream>
```

```
using namespace std;
class result2;
class result1 {
    private:
    string name;
    float marks;
    public:
     void accept() {
        cout<<"Enter student name:";
        getline(cin, name);
        cout<<"Enter the total marks obtained in
        first semester out of 100:";
        cin>>marks;
     }
     friend void average(result1, result2);
};
class result2 {
    private:
    float marks;
    public:
    void accept() {
        cout<<"Enter student name:";
        getline(cin, name)
        cout<<"Enter the total marks obtained
        in second semester out of 100:";
        cin>>marks;
    }
    friend void average(result1, result2);
};
```

```
void average(result1 x, result2 y){
    float avg;
    avg = (x.marks + y.marks) / 2;
    cout << "\nAverage of both the results=" <<
}
int main(){
    result1 r1;
    result2 r2;
    r1.accept();
    r2.accept();
    average(r1, r2);
    return 0;
}
```

Output:-

Enter student name: Kinjal Kolekar
Enter the total marks obtained in first semester out of 100: 98
Enter the total marks obtained in second semester out of 100: 96

Average of both the results = 97

5) WAP to find the greatest number among two numbers from two different classes using friend function.

```
#include <iostream>
using namespace std;
```

```
class num2;
class num1 {
    private:
    int a;
    public:
    void accept() {
        cout<<"Enter the first number:";
        cin>>a;
    }
    friend void grt(num1, num2);
};
class num2 {
    private:
    int b;
    public:
    void accept() {
        cout<<"Enter the second number:";
        cin>>b;
    }
    friend void grt(num1, num2);
};
void grt(num1 x, num2 y) {
    if(x.a>y.b) {
        cout<<x.a<<"is greater than"<<y.b;
    }
    else {
        cout<<y.b<<" is greater than "<<x.a;
    }
}
```

```
int main() {
    num1 n1;
    num2 n2;
    n1.accept();
    n2.accept();
    grt(n1, n2);
    return 0;
}
```

Output:-

Enter the first number: 8
Enter the second number: 10
10 is greater than 8.

FRIEND FUNCTION PRACTICE

1) Create two classes, Class A and Class B, each with a private. Write a friend function sum() that can acce private data from both classes and return the su

→

```
#include <iostream>
using namespace std;
class B;
class A {
    private:
    int a;
    public:
    void accept() {
        cout<<"Enter the first number:";
```

```
    cin>> a;
  }
  friend void sum(A, B);
};
class B {
  private:
  int b;
  public:
  void accept() {
      cout<< "Enter the second number:";
      cin>>b;
  }
  friend void sum(A, B);
};
void sum(A x, B y) {
    int sum;
    sum = x.a+y.b;
    cout<<"\nSum of the two numbers="<<sum;
}
int main() {
  A n1;
  B n2;
  n1.accept();
  n2.accept();
  sum(n1, n2);
}
```

Output:-

Enter the first number: 5

Enter the second number: 6

Sum of the two numbers = 11

2) Write a program with a class Number that contains private integer. Use a friend function swapNumbers (Number &, Number &) to swap the private values of two Number objects.

```
#include <iostream>
using namespace std;
class Number{
    private:
    int a, b;
    public:
    void accept(){
        cout<<"Enter the first number:";
        cin>>a;
    }
    void accept1(){
        cout<<"Enter the second number:";
        cin>>b;
    }
    void disp(){
        cout<<"\nAfter swapping:"<<endl;
        cout<<"First number="<<a<<endl;
    }
    void display(){
        cout<<"Second number="<<b;
```

```
    };
    friend void swap(Number &x, Number &y);
};
void swap(Number &x, Number &y){
        int temp;
        temp=x.a;
        x.a=y.b;
        y.b=temp;
}
int main(){
    Number n1;
    Number n2;
    n1.accept();
    n2.accept1();
    swap(n1, n2);
    n1.disp();
    n2.display();
    return 0;
}
```

Output:-

Enter the first number:7
Enter the second number:4

After swapping:
First number=4
Second number=7

3) Define two classes Box and Cube, each having a private

volume. write a friend function findGreater(Box, Cube) that determines which object has a larger volume.

→
```
#include <iostream>
using namespace std;
class Box;
class Cube {
    private:
    float volume;
    public:
    void accept() {
        cout << "Enter the volume of a cube:";
        cin >> volume;
    }
    friend void findgrt(Cube, Box);
};
class Box {
    private:
    float vol;
    public:
    void accept() {
        cout << "Enter the volume of a box:";
        cin >> vol;
    }
    friend void findgrt(Cube, Box);
};
void findgrt(Cube x, Box y) {
    if (x.volume > y.vol) {
```

```
    cout<<"\nCube has larger volume.";
  }
  else {
    cout<<"\nBox has larger volume.";
  }
}
int main() {
  Cube c;
  Box b;
  c.accept();
  b.accept();
  findgrt(c,b);
  return 0;
}
```

Output:-

Enter the volume of a cube: 567.88
Enter the volume of a box: 985.66

Box has larger volume.

4) Create a class Complex with real and imaginary parts as private members. Use a friend function to add two complex numbers and return the result as a new Complex object.

→

```
#include <iostream>
using namespace std;
class Complex{
```

```
private:
int r, i;
public:
void accept() {
    cout<<"Enter the real part:";
    cin>>r;
    cout<<"Enter the imaginary part:";
    cin>>i;
}
void disp() {
    cout<<r<<"+"<<i<<"i"<<endl;
}
friend Complex sum(Complex, Complex);
};
Complex sum(Complex x, Complex y) {
    Complex temp;
    temp.r = x.r + y.r;
    temp.i = x.i + y.i;
    return temp;
}
int main() {
    Complex c1, c2, c3;
    cout<<"Enter the first complex number:\n";
    c1.accept();
    cout<<"Enter the second complex number:\n";
    c2.accept();
    cout<<"\n# Complex Number 1:\n";
    c1.disp();
    cout<<"# Complex Number 2:\n";
```

```
    c2.disp();
    c3 = sum(c1, c2);
    cout << "\nSum of the two complex numbers=";
    c3.disp();
    return 0;
}
```

Output:-

Enter the first complex number:
Enter the real part: 5
Enter the imaginary part: 6
Enter the second complex number:
Enter the real part: 8
Enter the imaginary part: 2

Complex Number 1:
5+6i
Complex Number 2:
8+2i

Sum of the two complex numbers = 13+8i.

5) Create a class Student with private data members: name and three subject marks. Write a friend function calculateAverage(Student) that calculates and displays the average marks.

→
```
#include <iostream>
using namespace std;
```

```
class student {
        string name;
        int math;
        int phy;
        int chem;
        public:
        void accept() {
            cout << "Enter student name:";
            getline (cin, name);
            cout << "Enter marks obtained in maths:";
            cin >> math;
            cout << "Enter marks obtained in physics:";
            cin >> phy;
            cout << "Enter marks obtained in
            chemistry:";
            cin >> chem;
        }
        friend void calculateAverage(student);
};
void calculateAverage(student x) {
    float avg;
    avg = (x.math + x.phy + x.chem) / 3;
    cout << "\nAverage marks =" << avg;
}
int main() {
        student s;
        s.accept();
        calculateAverage(s);
        return 0;
}
```

Output:-
Enter student name: Kinjal Kolekar
Enter marks obtained in maths: 98
Enter marks obtained in physics: 96
Enter marks obtained in chemistry: 92

Average marks = 95

6) Create three classes: Alpha, Beta and Gamma, each with a private data member. Write a single friend function that can access all three and print their sum.

→
```
#include <iostream>
using namespace std;
class Beta;
class Gamma;
class Alpha{
    int a;
    public:
    void accept(){
        cout<<"Enter the first number:";
        cin>>a;
    }
    friend void sum(Alpha, Beta, Gamma);
};
class Beta{
    int b;
    public:
```

```
    void accept(){
        cout<<"Enter the second number:";
        cin>>b;
    }
    friend void sum(Alpha, Beta, Gamma);
};
class Gamma{
    int g;
    public:
    void accept(){
        cout<<"Enter the third number:";
        cin>>g;
    }
    friend void sum(Alpha, Beta, Gamma);
};
void sum(Alpha x, Beta y, Gamma z){
    int sum;
    sum = x.a+x.y+ y.b +z.g;
    cout <<"\n Sum ="<< sum;
}
int main (){
    Alpha a;
    Beta b;
    Gamma g;
    a.accept();
    b.accept();
    g.accept();
    sum(a, b, g);
    return 0;
}
```

Output:-
Enter the first number: 6
Enter the second number: 3
Enter the third number: 9

Sum = 18

7) Create a class point with private members x and y. Write a friend function that calculates and returns the distance between two Point objects.

→
```
#include <iostream>
using namespace std;
#include <math>
using namespace std;
class Point {
    int x, y;
    public:
    void accept() {
        cout << "Enter the x co-ordinate:";
        cin >> x;
        cout << "Enter the y co-ordinate:";
        cin >> y;
    }
    void disp() {
        cout << "(" << x << ", " << y << ")" << endl;
    }
    friend double diff(Point, Point);
};
```

```
double diff(Point x, Point y){
    double temp;
    return temp=sqrt(pow((x.x-y.x),2)+
    ((x.y-y.y),2));
}
int main(){
    Point p1, p2;
    cout<<"For the Point P:\n";
    p1.accept();
    cout<<" For the point Q:\n";
    p2.accept();
    cout<<"P=";
    p1.disp();
    cout<<"Q=";
    p2.disp();
    cout<<"\nDifference between the two
    points = "<<diff(p1,p2)<<"units";
}
```

Output:-

For the Point P:
Enter the x co-ordinate: 5
Enter the y co-ordinate: 6
For the Point Q:
Enter the x co-ordinate: 7
Enter the y co-ordinate: 8
P = (5,6)
Q = (7,8)

Difference between the two points = 2.82843 units.

8) Create two classes: BankAccount and Audit. BankAccount holds private balance information. Write a friend function in Audit that accesses and prints balance information for auditing.

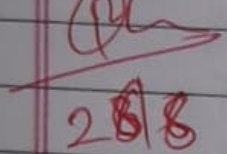
→
```
#include <iostream>
using namespace std;
class Audit;
class BankAccount {
    private:
    float balance;
    public:
    void accept() {
        cout<<"Enter the balance information:";
        cin>>balance;
    }
    friend class Audit;
};
class Audit {
    public:
    void disp (BankAccount x) {
        cout<<"Balance Information =" <<x.balance;
    }
};
int main() {
    BankAccount b;
    Audit a;
```

```
    b.accept();
    a.disp(b);
    return 0;
}
```

Output:-

Enter the balance information: 9879867.49

Balance information = 9867.49



EXPERIMENT-5

1) Write a program to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

→
```
#include <iostream>
using namespace std;
class Sum {
    int n, sum;
    public:
    Sum(int num) {
        n=num;
        sum=0;
        for(int i=0; i<n; i++) {
            sum+=i;
        }
        cout<<"Sum = "<<sum;
    }
};
int main() {
    Sum S(10);
    return 0;
}
```

Output:-

Sum=55

2) WAP to declare a class student having data

members as name and percentage. Write a constructor to initialize these data members and display data for one student.

→
```
#include <iostream>
#include <string>
using namespace std;
class student {
    private:
    string name;
    float percentage;
    public:
    student (string n, float p) {
        name = n;
        percentage = p;
        cout << "Name :";
        getline(cin, name);
        cout << "Percentage :";
        cin >> percentage;
    }
    void display() {
        cout << "student name :" << name << endl;
        cout << "Student percentage :" <<
        percentage << endl;
    }
};
int main() {
    student S1("Kinjal", 92.6);
    S1.display();
```

```
    return 0;
}
```

Output:-
Name: Kinjal
Percentage: 92.6
Student name: Kinjal
Student percentage: 92.6

3) Define a class college members variables as roll no, name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two objects of class and display the data.

→
```
#include <string>
#include <iostream>
using namespace std;
class College {
    private:
    string course_name, stud_name;
    int roll_no;
    public:
    College() {
        roll_no = 58;
        stud_name = "Kinjal";
        course_name = "AIDS";
    }
    void display() {
        cout << "Roll No:" << roll_no << endl;
```

```
        cout<< "Student Name:" << stud_name
        << endl;
        cout << "Course Name:" << course_name
        <<endl;
    }
};
int main() {
    celle College C1;
        C1.display();
        return 0;
}
```

Output:-

Roll No: 58
Student Name: Kinjal
Course Name: AIDS

4) WAP to demonstrate constructor overloading

→

```
#include <iostream>
using namespace std;
class rectangle {
    int l, b;
    public:
    rectangle (#) () {
        l=2;
        b=3;
    }
    rectangle (int x) {
        l=x;
```

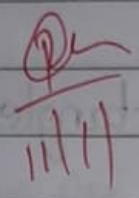
```
    b=x;
}
rectangle(int x, int y) {
        l=x;
        b=y;
}
rectangle(rectangle &r3) {
    l=r3.l;
    b=r3.b;
}
void calculate() {
    cout<<"The area is:"<<(l*b)<<endl;
}
};
int main() {
        rectangle r1;
        r1.calculate();
        rectangle r2(3);
        r2.calculate();
        rectangle r3(6,6);
        r3.calculate();
        rectangle r4(r3);
        r4.calculate();
        return 0;
}
```

Output:-

The area is: 6

The area is: 9

The area is: 36
The area is: 36



EXPERIMENT-6

1) WAP to implement multi-level inheritance. Assume suitable data.

→
```
#include <iostream>
#include <string>
using namespace std;
class Person {
        protected:
        int age;
        string name;
};
class student : public Person {
        private:
        int roll_no;
        public:
        void accept() {
        cout << "Enter your name:";
        cin>>name;
        cout<< "Enter your age:";
        cin>> age;
        cout << "Enter your roll no:";
        cin>>roll_no;
        }
        void display() {
        cout<< "Name:\n" <<name;
        cout<< "Age:\n" << age;
        cout<< "Roll No:\n" << roll_no;
        }
};
```

```
int main() {
    cout<<"Name:\n";
    Student s1;
    s1.accept();
    s1.display();
    return 0;
}
```

Output:-

```
Enter your name: Kinjal
Enter your age: 17
Enter your roll no: 58
Name: Kinjal
Age: 17
Roll No: 58
```

2) WAP to implement multiple inheritance. Assume suitable data.

→

```
#include <iostream>
using namespace std;
class Academic {
        protected:
        int marks;
};
class sports {
        protected:
        int score;
};
class Result: protected Academic, protected sports
```

```
int total_score=0;
public:
void accept() {
    cout<<"Enter the marks of the
    student:";
    cin>>marks;
    cout<<"Enter the sports score of the
    student:";
    cin>>score;
}
void calculate() {
    total_score =marks+score;
    cout<<"The marks of the student is:
    <<marks<<endl;
    cout<<"The sports score of the student
    is: <<score<<endl;
    cout<<"The total score of the
    student is:"<<total_score<<endl;
}
};
int main () {
    Result r;
    r.accept();
    r.calculate();
    return 0;
}
```

Output:-

Enter the marks of the student: 99

Enter the sports score of the student: 95
The marks of the student: 99
The sports score of the student: 95
The total score of the student: 194

3) WAP to implement hierarchical inheritance. Assume suitable data.

→
```
#include <iostream>
using namespace std;
class Vehicle {
        protected:
        string brand;
        int model;
};
class car:protected Vehicle {
        protected:
        string type;
};
class ElectricCar: protected car {
        int batteryCapacity;
        public:
        void accept() {
            cout<< "Enter the battery capacity of the car:";
            cin>>batteryCapacity;
        }
        void display() {
            cout<< "\nThe brand of the car:"<<
        void accept() {
```

```
cout cout << "Enter the brand of the car:";
    cin>> brand;
    cout<< "Enter the model of the car:";
    cin>>model;
    cout<< "Enter the type of the car:";
    cin>>type;
    cout << "Enter the battery capacity of
    the car:";
    cin>> batteryCapacity;
}
void display () {
    cout<< "\nThe brand of the car:"<<
    brand;<<endl;
    cout<< "The model of the car:"<<
    model<<endl;
    cout << "The type of the car:"<<
    type <<endl;
    cout<< "The battery capacity of the
    car:" <<batteryCapacity;<<endl;
}
};
int main () {
    ElectricCar e;
    e.accept();
    e.display();
    return 0;
}
```

Output:-

Enter the brand of the car: BMW
Enter the model of the car: M4
Enter the type of the car: Sedan Sport
Enter the battery capacity: 115
The brand of the car: BMW
The model of the car: M4
The type of the car: Sedan Sport
The battery capacity of the car: 115

4) WAP to implement hybrid inheritance.

```
#include <iostream>
#include <string>
using namespace std;
class Person {
    public:
    string name;
    int age;
    void getPersonDetails() {
        cout<<"Enter name: ";
        cin>>name;
        cout<<"Enter age: ";
        cin>>age;
    }
    void showDetails() {
        cout<<"Name: "<<name<<"Age: "<<
        age<<endl;
    }
};
class Student: public Person {
```

```
public:
string course;
void getDetails() {
    cout<<"Enter course:";
    cin>>course;
}
void showDetails() {
    cout<<"Course:"<<course<<endl;
}
};
class Employee:public Person {
        public:
        string company;
        void getEmployeeDetails() {
            cout<<"Enter company:";
            cin>>company;
        }
        void showEmployeeDetails() {
            cout<<"Company:"<<company<<endl;
        }
};
class Intern:public Student, public Employee {
    public:
    void showInternDetails() {
        cout<<"\n----Intern Details----";
        student::Show showDetails();
        showEmployeeDetails();
    }
};
```

```
int main () {
    Intern it;          Person
    it.Student::getDetails();
    it.getDetails();
    it.getEmployeeDetails();
    it.showInternDetails();
    return 0;
}
```

Output:-

```
Enter Name: Kinjal
Enter age: 17
Enter course: Computer Science
Enter company: Microsoft

----Intern Details----
Name: Kinjal  Age: 17
Course: Computer Science
Company: Microsoft
```

EXPERIMENT-7

1) WAP using function overloading to calculate the area of a laboratory (which is rectangle in shape) and area of classroom.

→
```
#include <iostream>
using namespace std;
class Area {
    private:
    float l,b;
    public:
    void area(float l) {
        float area;
        area = l * l;
        cout << "Area of square:" << area << endl;
    }
    void area(float l, float b) {
        float area;
        area = l * b;
        cout << "Area of rectangle:" << area;
    }
};
int main() {
    Area a1;
    a1.area(8);
    a1.area(4,12);
    return 0;
}
```

Output:-
Area of square: 64
Area of rectangle: 48

2) WAP using function overloading to calculate the sum of 5 float values and 10 integers.

→
```
#include <iostream>
using namespace std;
Class sum {
    private:
    int a, b, c, d, e, f, g, h, i, j;
    float k, l, m, n, o;
    public:
    void Sum(float k, float l, float m, float n,
    float o) {
        float Sum;
        Sum = k+l+m+n+o;
        cout<<"Sum of 5 floating numbers:"
        <<sum<<endl;
    }
    void Sum(int a, int b, int c, int d, int e,
    int f, int g, int h, int i, int j) {
        int sum;
        sum = a+b+c+d+e+f+g+h+i+j;
        cout<< "Sum of 10 integers:" << sum;
    }
};
int main() {
```

```
    Sum s1;
    s1.sum(1.2,3.5,5.6,8.4,10.5);
    s1.sum(1,2,3,4,5,6,7,8,9,10);
    return 0;
}
```

Output:-
Sum of 5 floating numbers: 20.2
Sum of 10 integers: 55

3) WAP to implement unary operator when used with the object so that the numeric data member of the class is negated.

→
```
#include <iostream>
using namespace std;
class Number {
    private:
    int x;
    public:
    void accept() {
        cout<<"Enter a number: ";
        cin>>x;
    }
    void operator-() {
        x=-x;
    }
    void display() {
        cout<<"Negated number: "<<x<<endl;
```

```
    }
};
int main() {
    Number n1;
    n1.accept();
    -n1;
    n1.display();
    return 0;
}
```

Output:-
Enter a number: 5
Negated number: -5

4) WAP to implement the unary ++ operator (for pre-increment and post increment) when used with the object so that the numeric data member of the class is incremented.

→
```
#include <iostream>
using namespace std;
class Number {
    private:
    int x;
    int accept;
    public:
    void accept() {
        cout << "Enter a number:";
        cin >> x;
```

```
    temp=x;
}
void operator++(int) {
    x=x++;
}
void display1() {
    cout<<"(pre) The number is:"<<x<<endl;
}
void display2() {
    cout<<"(post) The number is:"<<x;
}
};
int main() {
    Number n1;
    n1.accept();
    ++n1;
    n1.display1();
    n1.reset();
    n1++;
    n1.display2();
    return 0;
}
```

Output:-

Enter a number:5

(pre) The number is:6

(post) The number is:5

EXPERIMENT-8

1) WAP to overload '+' operator so that two strings can be concatenated. Eg. 'XYZ' + 'PQR' = 'XYZPQR'

→
```
#include <iostream>
#include <string>
using namespace std;
class MyString {
    public:
    string str;
    MyString operator + (MyString &other) {
        MyString temp;
        temp.str = str + other.str;
        return temp;
    }
    void display() {
        cout << str << endl;
    }
};
int main() {
    MyString s1, s2, s3;
    s1.str = 'XYZ';
    s2.str = 'PQR';
    s3 = s1 + s2;
    cout << "Concatenated string: ";
    s3.display();
    return 0;
}
```

Output:-
Concatenated string: XYZPQR

2) #include <iostream>
#include <string>
using namespace std;
class login {
protected:
string name, password;
public:
virtual void accept() {
cout << "Enter name:";
cin >> name;
cout << "Password:";
cin >> password;
}
};
class EmailLogin : Public Login {
string email;
public:
void accept() & override {
Login::accept();
cout << "Enter email:";
cin >> email;
}
void display() {
cout << "\n---- Email Login Details ---
-\n";
cout << "Name:" << name << "Password:"

```
<<password<<"Email:"<<email<<endl;
    }
};
class MembershipLogin:public Login {
        string membershipID;
        public:
        void accept() override {
            Login::accept();
            cout<<"Enter membership ID:";
            cin>>membershipID;
        }
        void display(){
            cout<<"\n---Membership Login---
            \n";
            cout<<"Name:"<<name<<"\n
            Password:"<<password<<"Members
            ID:"<<membershipID<<endl;
        }
};
int main() {
    EmailLogin e;
    MembershipLogin m;
    e.accept();
    m.accept();
    e.display();
    m.display();
    return 0;
}
```

Outputs-

Enter Name: Kinjal
Enter password: mitupu@123
Enter email: abc@gmail.com
Enter name: Kinjal
Password: 745
Enter membership ID: 5745
---- Email Login Details ----
Name: Kinjal
Password: mitupu@123
Email: abc@gmail.com
--- Membership Login Details ---
Name: Kinjal
Password: 745
Membership ID: 5745

11/11

EXPERIMENT-9

1) WAP to copy contents of one file into another.

→
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
    fstream new_file;
    fstream new_file_write;
    new_file.open("new_file", ios::in);
    new_file_write.open("new_file_write.txt",
    ios::out);
    if(!new_file) {
        cout<<"Failed to open input file.";
    }
    else if(!new_file_write) {
        cout<<"Failed to open output file.";
    }
    else {
        cout<<"Files opened successfully.";
        string line;
        while(getline getline(new_file, line))
        new_file_write<<line<<endl;
        new_file.close();
        new_file_write.close();
    }
    return 0;
}
```

2) WAP to count digits and spaces using file handling.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
    fstream new_file;
    new_file.open("new_file", ios::in);
    if (!new_file) {
        cout<< "Failed to open the file.";
        char ch;
        int digitcount =0;
        int spacecount =0;
        while (new_file.get(ch)) {
            if(isdigit(ch)) {
                digitcount++;
            }
            if(isspace(ch)) {
                spacecount++;
            }
    } }
    cout<< "\nTotal Digits =" <<digitcount<<endl;
    cout<< "Total spaces =" << space count;
    new_file.close();
}
    return 0;
}
```

3) WAP to count words using file handling.

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
int main () {
    fstream new_file;
    new_file.open ("first.txt", ios::in);
    if (!new_file) {
        cout << "Error occured!" << endl;
        return 1;
    }
    char ch;
    int words_count = 0;
    bool inword = false;
    while (new_file.get(in)) {
        if (isspace(ch)) {
            inword = false;
        }
        else if (!inword) {
            word_count++;
            inword = true;
        }
    }
new_file.close();
cout << "No. of words: " << words_count << endl;
return 0;
}
```

4) WAP to count occurrence of a given word.

→
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
    fstream new_file;
    new_file.open("first.txt", ios::in);
    if(!new_file) {
        cout<<"Error in opening first.txt!"<<endl;
        return 1;
    }
    string target = "Kinjal";
    string word;
    int count=0;
    while(new_file>>word) {
        if(word==target) {
            count++;
        }
    }
    new_file.close();
    cout<<"The word"<<target<<" occured"<<
    count<<"times"<<endl;
    return 0;
}
```

11/11

EXPERIMENT-10

1)
```
#include <iostream>
using namespace std;
template <typename T>
void SumArr(T arr[], int n) {
    T sum=0;
    for(int i=0; i<n; i++) {
        sum+=arr[i];
    }
    cout<<"Sum is:"<<sum<<endl;
}
int main() {
    int n=7;
    int arr[n]={1,2,3,4,5,6,7};
    float arr1[n]={1.01,1.02,1.03,1.04,1.05,1.06,1.07};
    double arr2[n]={1.89465, 2.94865, 3.84653,
    4.78456, 5.48966, 6.46853, 7.21564};
    SumArr(arr,n);
    SumArr(arr1,n);
    SumArr(arr2,n);
    return 0;
}
```

Output:-
Sum is: 28
Sum is: 9.8
Sum is: 32.6482

2)
```
#include <iostream>
```

```
using namespace std;
template <typename T>
T squr (T n) {
    return n*n;
}
template <>
string squr <string>(string str) {
    return str+str;
}
int main () {
    cout<< squr <int> (3) <<endl;
    cout<< squr <string>("Hello");
}
```

Output:-

9
HelloHello

3)
```
#include <iostream>
using namespace std;
template <typename t1, typename t2>
class cal {
    t1 n1;
    t2 n2;
    public:
    void acc () {
        cout<< "Enter N1 and N2:";
        cin>> n1>> n2;
    }
```

```
void calc(int choice) {
    switch(choice) {
        case 1:
            cout<<"Addition:"<<n1+n2;
            break;
        case 2:
            cout<<"Substraction:"<<n1-n2;
            break;
        case 3:
            cout<<"Multiplication:"<<n1*n2;
            break;
        case 4:
            cout<<"Division:"<<n1/n2;
            break;
        default:
            cout<<"Invalid syntax!";
    }
};
int main() {
    int choice;
    cout<<"1. Addition \n 2. Substraction \n 3. Multiplication
    \n 4. Division \n";
    cout<<"Enter your choice:";
    cin>>choice;
    cal <float, int> a;
    a.acc();
    a.calc(choice);
    return 0;
}
```

Output:-

1. Addition
2. Substraction
3. Multiplication
4. Division

Enter your choice: 1
Enter N1 and N2: 2 3
Addition: 5

4) #include <iostream>
using namespace std;
template <typename T>
class stack {
    T arr[100];
    int top;
    public:
    stack() {top=-1;}
    void push (T val) {
        if (top==99) {
            cout<<"Stack overflow"<<endl;
        }
        else {
            arr[++top]=val;
            cout<<val<<" pushed into stack"<<endl;
        }
    }
    void pop() {
        if(top==-1) {
            cout<<"Stack underflow";

```
        }
        else {
            cout<<arr[top--]<<" popped from
            stack"<<endl;
        }
    }
    void display() {
        if(top==-1) {
            cout<<"Stack is empty:"<<endl;
        }
        else {
            cout<<"Stack elements:";
            for(int i=top; i>=0; i--) {
                cout<<arr[i]<<" ";
            }
            cout<<endl;
        }
    }
};
int main() {
    stack <int> s;
    int choice, val;
    do {
        cout<<"\n---Stack Menu---\n";
        cout<<"1.Push\n2.Pop\n3.Display\n4.Exit\n";
        cout<<"Enter your choice:";
        cin>>choice;
        switch(choice) {
            case 1:
```

```
            cout<<"Enter value to push:";
            cin>>val;
            s.push(val);
            break;
        case 2:
            s.pop();
            break;
        case 3:
            s.display();
            break;
        case 4:
            cout<<"Exiting....."<<endl;
            break;
        default:
            cout<<"Invalid choice!"<<endl;
    }
  }
  return 0;
}
```

Output:-

```
---Stack Menu---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter value to push: 10
10 pushed in stack
Menu....
```

Enter your choice: 1
Enter value: 20
20 pushed in stack

Menu....
Enter your choice: 1
Enter value: 30
30 pushed in stack.

Menu....
Enter your choice: 3
~~Enter~~ Stack elements: 30 20 10

Menu....
Enter your choice: 2
30 popped from stack

Menu....
Enter your choice: 3
Stack elements: 20 10

Menu....
Enter your choice: 4
Exiting....

EXPERIMENT-11

1)
```
#include <bits/stdc++.h>
using namespace std;
class vect {
    public:
    vector<int> vec = {10,20,30,40,50,60};
    void modify () {
        int idx, val;
        cout<<"Enter pos:";
        cin>>idx;
        cout<<"Enter values";
        cin>>val;
        for(int i=0
```

1)
```
#include <iostream>
#include <vector>
using namespace std;
int main () {
    vector <int>V = {1,2,3,4,5,6,7,8,9,10};
    cout<<"Initial vector:"<<endl;
    for(int i=0; i<10; i++) {
        cout<<v[i]<<" "<<endl;
    }
    cout<<"multiply by 10"<<endl;
    for(int i=0; i<10; i++) {
        v[i] = v[i] *10;
    }
    cout<<"New vector:"<<endl;
    for(int i=0; i<10; i++) {
```

```
        cout<<"v[i] <<" " <<endl;
    }
    return 0;
}
```

2)
```
#include <iostream>
#include <vector>
using namespace std;
int main () {
    vector<int> v = {1,2,3,4,5,6,7,8,9,10};
    cout<<"Initial vector:" <<endl;
    for(vector <int> ::iterator it = v.begin();
    it != v.end(); ++it) {
        cout<<*it <<" ";
    }
    cout<<endl;
    cout<<"Multiply by 10" <<endl;
    for(vector<int>::iterator it = v.begin();
    it != v.end(); ++it) {
        *it = (*it) * 10;
    }
    cout <<"New vector:" <<endl;
    for(vector<int> ::iterator it = v.begin();
    it != v.end(); ++it) {
        cout <<*it <<" ";
    }
    cout <<endl;
    return 0;
}
```

EXPERIMENT-12

1) WAP to implement stack.

```
#include <iostream>
#include <stack>
using namespace std;
stack <int> stack1;
void disp() {
        stack <int> temp=stack1;
        while(!temp.empty()) {
          cout<<temp.top()<<" ";
          temp.pop();
        }
        cout<<endl;
}
int main () {
    int num;
    cout<<"Enter a no.:"<<endl;
    cin>>num;
    for(int i=0;i<num;i++) {
        int temp;
        cout<<"Enter element at position:"<<i+1<<
        endl;
        cin>>temp;
        stack1.push(temp);
    }
    cout<<endl;
    cout<<"Top most element:"<<stack.top()<<
    endl;
```

```
cout << "Stack elements(top to bottom):"<<
stack.disp() <<endl;
cout<<"Top Pop function:" <<endl;
stack1.pop();
disp();
stack1.pop();
disp();
stack1.pop();
disp();
}
```

2) WAP to implement STL Queue.

→
```
#include <iostream>
#include <queue>
using namespace std;
int main () {
    queue <int> q;
    for (int i=0; i<=10; i++) {
        q.push(i*10);
    }
    cout <<endl;
    cout << "Front element:" << q.front() <<endl;
    cout<< "Back element:" << q.back() <<endl;
    cout <<endl;
    q.pop();
    cout<< "After one pop front=" << q.front()<<
    endl;
    cout<<endl;
```

```
cout << "Queue elements (front to back): ";
while (!q.empty()) {
    cout << q.front() << " ";
}
}
```